polyturtle

olivier_pirson_opi@yahoo.fr
http://www.opimedia.be/

dimanche 10 juillet 2011

Résumé

Utilisation de la tortue graphique en Python, avec une version en PostScript. Suivi d'un petit exemple d'utilisation de Graphviz à partir de Python. (Les différents fichiers utilisés sont disponibles en ligne : http://www.opimedia.be/DS/petits_papiers/ . Ces programmes sont compatibles avec Python 3.)

1 La tortue graphique en Python

Le module **standard** `turtle`[1] implémente en Python[2] la tortue graphique jadis popularisée par le langage Logo. (Un module **optionnel** `xturtle`[3] offre un peu plus de contrôle sur la tortue.)
Le livre en ligne *How to Think Like a Computer Scientist: Learning with Python*[4] est une bonne initiation.

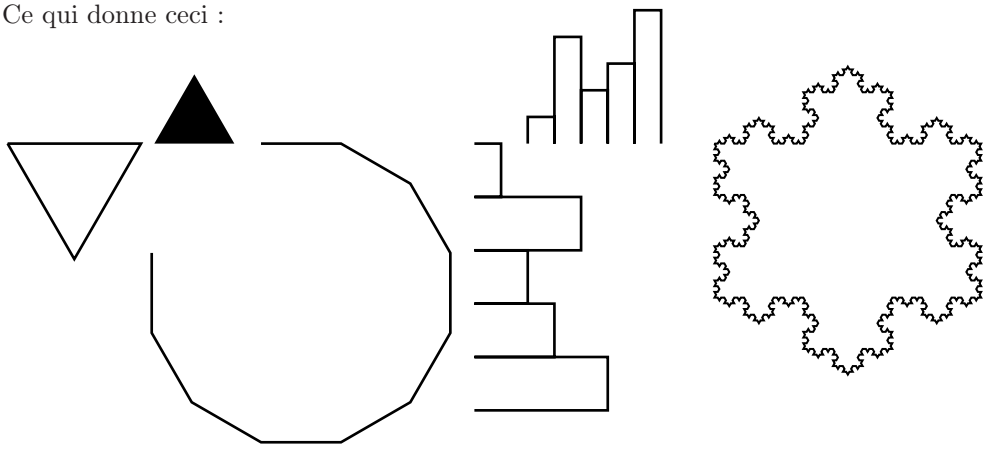
```
#!/usr/bin/env python
# -*- coding: latin-1 -*-
```

```
from turtle import *

def flock(len=100, ite=4):
    """Trace (à partir de la position courante) un flocon de von Koch"""
    for i in range(0, 3):
        flock_side(len=len, ite=ite)
        right(120)

def flock_side(len=100, ite=4):
    """Trace (à partir de la position courante) un côté d'un flocon de von Koch"""
    if ite > 0:
        flock_side(len=len/3.0, ite=ite - 1)
        left(60)
        flock_side(len=len/3.0, ite=ite - 1)
        right(120)
        flock_side(len=len/3.0, ite=ite - 1)
        left(60)
        flock_side(len=len/3.0, ite=ite - 1)
    else:
        forward(len)

def histo(l, w=10):
    """Trace (à partir de la position courante) un histogramme de largeur w
    avec les éléments de la séquence l"""
    if len(l) > 0:
        left(90)
        forward(l[0])
        right(90)
        forward(w)
        right(90)
        forward(l[0])
        left(90)
        histo(l[1:], w=w)

def poly(c, len=50, nb=None, anti=False):
    """Trace dans le sens horloger (à partir de la position courante)
    nb côtés d'un polygône régulier à c côtés.
    Si nb == None alors trace les c côtés
```

1. http://docs.python.org/library/turtle.html#module-turtle
2. http://www.python.org/
3. http://xturtle.rg16.at/
4. http://www.openbookproject.net/thinkcs/

```
    Si anti alors trace dans le sens anti-horloger"""
    if nb == None:
        nb = c
    if nb > 0:
        forward(len)
        if anti:
            left(360.0/c)
        else:
            right(360.0/c)
        poly(c, len=len, nb=nb - 1, anti=anti)

def to(x, y):
    """Se déplace sans tracer à la position (x, y) et fixe l'orientation à 0"""
    up()
    goto(x, y)
    setheading(0)
    down()


#tracer(False)  # pour désactiver le mode tortue au ralenti
delay(1)  # rapidité de la tortue (en ms)

to(-200, 0)
poly(3)

to(-140, 0)
begin_fill()
poly(3, len=30, anti=True)
end_fill()

to(-100, 0)
poly(12, len=30, nb=10)

to(-20, 0)
right(90)
histo([10, 40, 20, 30, 50], w=20)

to(0, 0)
histo([10, 40, 20, 30, 50])

to(70, 0)
flock()

done()  # pour garder la fenêtre ouverte
```

Ce qui donne ceci :



En fait cette image a été générée (pour l'inclure dans ce document) par le programme PostScript suivant...

2 Version PostScript

Un programme en PostScript [5] qui produit les mêmes dessins que le programme Python qui précède.

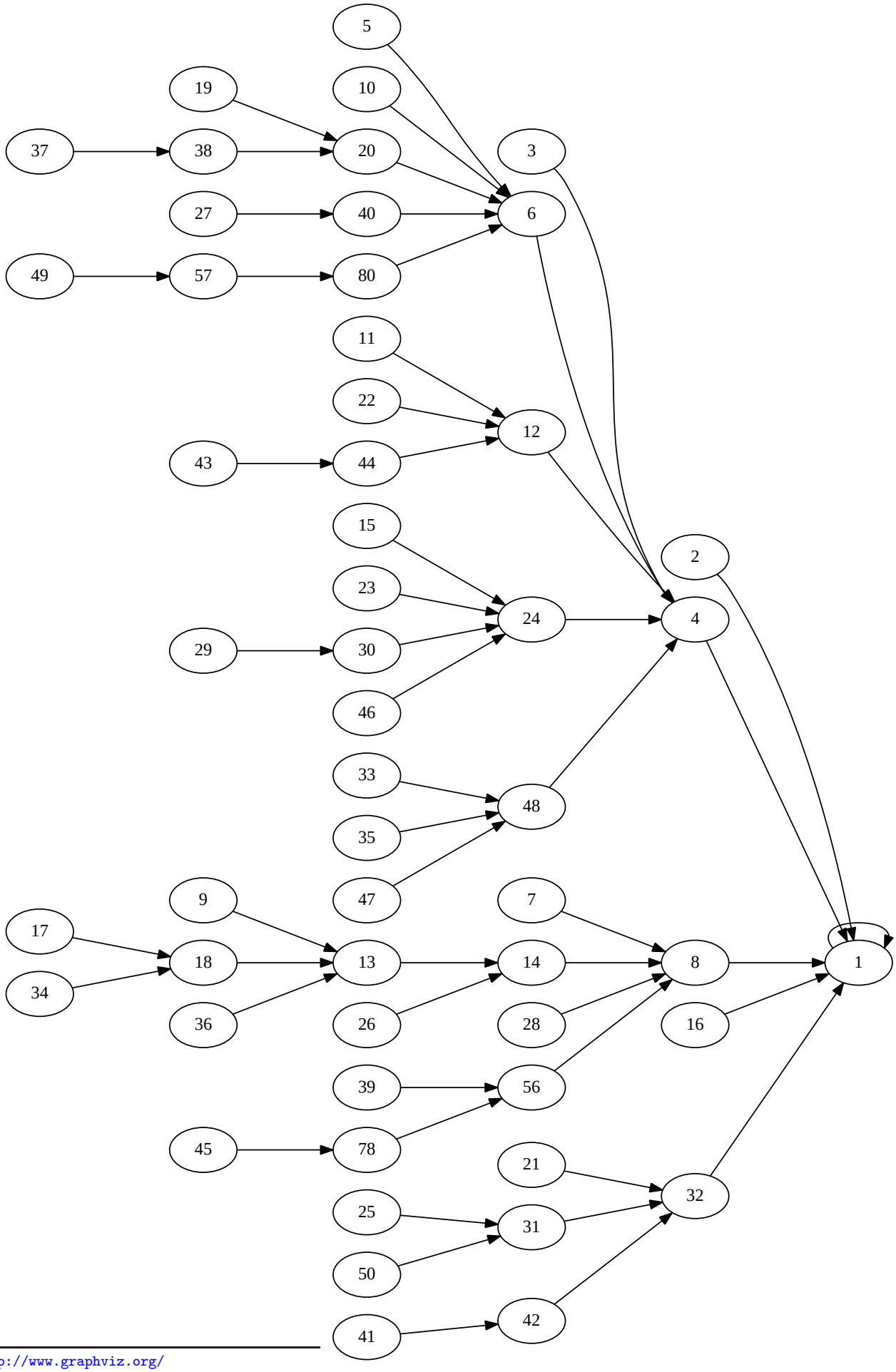
5. Interpréteur PostScript : GhostScript, http://www.ghostscript.com/

```
%!PS-Adobe-3.0 EPSF-3.0
%%BoundingBox: 3 86 372 251
%%EndComments

save

% Fonctions
%%%%%%%%%%%
/forward {% longueur
    0 rlineto
} def

/left {% angle
    rotate
} def

/right {% angle
    neg rotate
} def


/flock {% len, ite
    /ite exch def
    /len exch def

    len ite
    len ite
    len ite flock_side
    120 right
    flock_side
    120 right
    flock_side
    120 right
} def

/flock_side {% len, ite
    /ite exch def

    ite 0 gt {
        /len exch 3 div def
        /ite ite 1 sub def

        len ite
        len ite
        len ite
        len ite flock_side
        60 left
        flock_side
        120 right
        flock_side
        60 left
        flock_side
    } {
        forward
    } ifelse
} def

/histo {% -1, hk, ..., h1, w
    /w exch def
    /h exch def

    h 0 ge {
        90 left
        h forward
        90 right
        w forward
        90 right
        h forward
        90 left
```

```
        w histo
    } if
} def

/poly {% c, len
    /len exch def
    /c exch def

    len forward
    360 c div
    anti {left} {right} ifelse
    c len c 1 sub poly_nb
} def

/poly_nb {% c, len, nb
    /nb exch def
    /len exch def
    /c exch def

    nb 0 gt {
        len forward
        360 c div
        anti {left} {right} ifelse
        c len nb 1 sub poly_nb
    } if
} def


% Variable globale
%%%%%%%%%%%%%%%%%%%
/anti false def  % utilisée par poly et poly_nb pour fixer ou non le sens anti-horloger


% main
%%%%%%%
5 200 moveto
3 50 poly
stroke  % effectue le tracé

60 200 moveto
/anti true def
3 30 poly
/anti false def
fill  % effectue le tracé et remplit la figure

100 200 moveto
12 30 10 poly_nb
stroke

60 right
180 200 moveto
90 right
-1 50 30 20 40 10 20 histo
stroke

90 left
200 200 moveto
-1 50 30 20 40 10 10 histo
stroke

.75 setlinewidth
270 200 moveto
100 4 flock
stroke

showpage

restore
%%EOF
```

3 Petit exemple d'utilisation de Graphviz à partir de Python

L'ensemble des logiciels `dot` [6] permet de facilement réaliser des graphes tels que celui-ci :



6. http://www.graphviz.org/

Ce graphe a été généré à partir du fichier `.dot` suivant :

```
digraph "sigma odd" {
  size="7, 10";
  margin="0";
  ranksep=".2";
  nodesep=".2";
  rankdir="LR";
  ratio="fill";

  1 -> 1;
  2 -> 1;
  3 -> 4;
  4 -> 1;
  5 -> 6;
  6 -> 4;
  7 -> 8;
```

...

```
  56 -> 8;
  57 -> 80;
  78 -> 56;
  80 -> 6;
}
```

Qui a lui-même été généré à partir du petit programme Python :

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-
```

```
import DSPython.factors as factors  # cf. http://www.opimedia.be/DS/DSPython/

# Génère dans un dictionnaire la liste des arcs du graphe
d = {}
d[1] = 1
for n in range(2, 51):
    s = n
    while s >= n:
        d[s] = factors.divisors_sum_odd(factors.primaries(s))
        s = d[s]

# Envoye sur la sortie standard le contenu du fichier .dot
print("""digraph "sigma odd" {
  size="7, 10";
  margin="0";
  ranksep=".2";
  nodesep=".2";
  rankdir="LR";
  ratio="fill";
""")
for n in d:
    print('  {0} -> {1};'.format(n, d[n]))
print('}')
```

Le graphe représente les parcours $n, \sigma_{impair}(n), \sigma^2_{impair}(n), \sigma^3_{impair}(n) \ldots$ pour les n allant de 1 à 50.

Je **conjecture** que $\forall n \in \mathbb{N}_*, \exists k \in \mathbb{N} : \sigma^k_{impair}(n) = 1$ (cf. le problème σ_{impair})
et $\forall n \in \mathbb{N}_* : \sigma^n_{impair}(n) = 1$

Table des matières